



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/955,764	09/19/2001	Jun Li	10007965	9833

7590 04/02/2007
HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, CO 80527-2400

EXAMINER

RUTTEN, JAMES D

ART UNIT	PAPER NUMBER
----------	--------------

2192

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
2 MONTHS	04/02/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.



UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

MAILED

Application Number: 09/955,764
Filing Date: September 19, 2001
Appellant(s): LI ET AL.

APR 02 2007

Technology Center 2100

Tiep H. Nguyen, Reg. No. 44,465
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 12/26/2006 appealing from the Office action mailed 06/06/2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal (a-d) is correct.

Art Unit: 2192

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

Kazi et al. "JaViz: A client/server Java profiling tool" Java Performance, Vol. 39, No. 1 (2000),
18 pages

Blumson et al. "Automatic Insertion of Performance Instrumentations for Distributed
Applications" Center for Information Technology Integration, University of Michigan,
Technical Report (February 1995)

Peek et al. "Unix Power Tools" O'Reilly, 2nd Edition (August, 1997), Chapter 38, section 5

4,819,233	DELUCIA	4-1989
6,151,639	TUCKER ET AL	11-2000
5,522,073	COURANT ET AL	5-1995
5,146,593	BRANDLE ET AL	9-1992

(9) Grounds of Rejection

The following grounds of rejection are applicable to the appealed claims:

- A. Claims 1-7, 9-11, 13-19, 21-32, and 35-42 are rejected under 35 U.S.C. 103(a) as
being unpatentable over prior art of record "JaViz: A client/server Java profiling tool"
by Kazi et al. (hereinafter referred to as "Kazi") in view of prior art of record

Art Unit: 2192

“Automatic Insertion of Performance Instrumentation for Distributed Applications”
by Blumson et al. (hereinafter referred to as “Blumson”) in view of prior art of record
U.S. Patent 4,819,233 to Delucia et al. (hereinafter “Delucia”), in view of prior art of
record U.S. Patent 6,151,639 to Tucker et al. (hereinafter “Tucker”). *[See herein
pages 4-16]*

As per claim 1, Kazi discloses:

*A monitoring method for a component-based software system operating
over one or more processing devices* See Kazi page 1, Abstract:

The JaViz performance analysis tool generates
execution traces with sufficient detail to determine
program hot spots, **including remote method calls**, in
a **distributed Java application** program

also page 8 paragraph 3:

...executing on a physically distributed **processor**.

comprising the steps of:

*initiating an invocation of a second software component from within an
execution of a first software component* See Kazi page 8 paragraph 3 under

“Client/server trace generation”:

The Java remote method invocation (RMI) facility
allows **one Jvm to execute a method on another Jvm**,
which may be executing on a physically distributed
processor.

recording a stub start log data including a global causal identifier

<within microseconds of> said invocation of said second software component

See Kazi page 7 last paragraph under “Detailed trace generation”:

Art Unit: 2192

The trace generation module of the **Jvm** is modified to **record every invocation** of a method using time stamps that **show the start and end times** of the method with microsecond resolution.

also page 5 paragraph 3:

In addition to the parent-child links to reflect the call graph, each record contains such information as the number of methods invoked by this method, the time when the method started, the time when it completed, the thread executing this method, the **method identifier** of the method call being represented, and the specific **Jvm** on which the method is executed.

wherein the second software component executes on a separate thread and in a system remote from the first software component; See fourth full paragraph on page 8:

For every **remote method** invoked through RMI, JaViz's modified **Jvm** records these identifiers at both the client side and the server side.

Note that this passage shows that a remote component is invoked, which must inherently execute in a separate thread since a remote system cannot execute a local thread.

recording a stub end log data including the global causal identifier in said instrumented stub after a response is received from said invocation of said second software component, said response including the global causal identifier See Kazi page 7 last paragraph as cited above.

wherein said stub start log data and said stub end log data gather runtime information about execution of said second software component within said component-based software system See Kazi page 7 last paragraph as cited above.

Kazi does not expressly disclose an instrumented stub, recording log data *before* invocation, or *transmitting the global causal identifier from the first software component to the second software component.*

However, in an analogous environment, Blumson teaches instrumenting a stub to collect runtime data. See page 6, Section 6.1: “Our IDL compiler has an additional command-line flag...to **insert instrumentation.**”

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Blumson’s stub instrumentation implementation in Kazi’s instrumented jvm. One of ordinary skill would have been motivated to take measurements on certain operations such as marshalling time that are otherwise difficult, while maintaining a relatively simple implementation versus modification of a runtime library.

While Kazi discloses that log data is recorded with microsecond resolution (Kazi page 7 last paragraph under “Detailed trace generation”), Kazi does not expressly disclose whether this data is recorded before or after invocation.

However, in an analogous environment, Delucia teaches that data can be recorded before and after invocation. See column 2 lines 37-42:

Where the target code unit calls another routine, executable write instructions are inserted by a processor before and after the call statement to generate in the output documentation an indication that the call statement was reached and that the program returned to the correct location after the call.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Delucia’s teaching of code instrumentation techniques

with Kazi's instrumented JVM. One of ordinary skill would have been motivated to place instrumentation before and after a call as an indication that the call completed successfully (Delucia column 2 lines 41 and 42).

Kazi discloses that global method identifiers are used at both the server and the client. See page 9 3rd paragraph:

The client entry indicates that it is an RMI call to **Object 25 for Method 5** on the server... The corresponding entry in the server profile indicates that it is an incoming RMI call from the client on Machine csInt3.cs.umn.edu through Port 4667 for **Method 5 on Object 25**.

Kazi does not expressly disclose how the same identifiers appear at both the client and the server. However, in an analogous environment, Tucker teaches that identifiers can be transmitted between remote machines. See column 3 lines 22-24:

The system-wide identifier is transmitted in a remote object invocation request to the appropriate remote node 102b.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Tucker's teaching of transmitting identifiers with Kazi's identifiers. One of ordinary skill would have been motivated to transmit identifiers when a network connection is available, but a shared file system is not present (Tucker column 1 lines 14-17).

As per claim 2, the above rejection of claim 1 is incorporated. Kazi does not expressly disclose generation of a stub.

However, Blumson further discloses: *wherein said instrumented stub is generated from a description of an interface of said second software component* (page 5 paragraph 1; also page 6 Section 6.1 as cited above).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to generate a stub from a description of an interface. One of ordinary skill would have been motivated to generate a stub according to standard practices of using the Open Software Foundation's Distributed Computing Environment (DCE) based on a remote procedure call (RPC) paradigm.

As per claim 3, the above rejection of claim 1 is incorporated. Kazi further discloses: *wherein said second software component is remote from said first software component* (page 8 paragraph 3 as cited above).

As per claim 4, the above rejection of claim 1 is incorporated. Kazi further discloses: *wherein said first software component resides on a first processing device and said second software component resides on a second processing device* (page 8 paragraph 3 as cited above).

As per claim 5, the above rejection of claim 1 is incorporated. Kazi further discloses: *the preliminary step of selecting a log data contents to be included in said stub start and stub end log data, with the selecting step logging*

zero or more of an application semantic behavior data, a timing latency data, a shared resource usage data, and a causality relationship data (page 4 paragraph 3).

As per claim 6, the above rejection of claim 1 is incorporated. Kazi does not expressly disclose configuration during generation of the stub.

However, Blumson teaches: *wherein a log data contents is configured during generation of said instrumented stub* (page 6 Section 6.1 paragraph 1: Flags and subflags).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Blumson's flags for stub generation in Kazi's instrumented jvm. One of ordinary skill would have been motivated to minimize perturbation of a target system by inserting only the necessary amount of instrumentation code, thereby reducing the time spent executing logging procedures.

As per claim 7, the above rejection of claim 1 is incorporated. Kazi further discloses: *wherein a log data contents is configured during operation of said component-based software system* (page 5 paragraph 5: "Visualizer").

As per claim 9, the above rejection of claim 1 is incorporated. Kazi further discloses:

initiating said invocation of said second software component from within an execution of a skeleton (page 8 paragraph 3 as cited above);

recording a skeleton start log data before said skeleton invokes said second software component (page 7 last paragraph as cited above); and

recording a skeleton end log data in said skeleton after a response is received from said invocation of said second software component (page 7 last paragraph as cited above).

Kazi does not expressly disclose an instrumented skeleton.

However, Blumson teaches instrumenting server “stubs” or skeletons to collect runtime data (page 6, Section 6.1).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Blumson’s skeleton instrumentation implementation in Kazi’s instrumented jvm. One of ordinary skill would have been motivated to take measurements on certain operations such as marshalling time that are otherwise difficult, while maintaining a relatively simple implementation versus modification of a runtime library.

As per claims 10 and 11, the above rejection of claim 9 is incorporated.

All further limitations have been addressed in the above rejections of claims 2 and 3, respectively.

As per claim 13, the above rejection of claim 9 is incorporated. Kazi further discloses: *wherein an accumulated log data from a plurality of instrumented stubs and a plurality of instrumented skeletons is collected and correlated* (Page 3 under “The JaViz performance visualization tool set”).

As per claim 14, the above rejection of claim 9 is incorporated. Kazi further discloses: *wherein said stub start, stub end, skeleton start, and skeleton end log data capture a causality relationship data between said first software component and said second software component* (Page 3 under “The JaViz performance visualization tool set”).

As per claim 15, the above rejection of claim 9 is incorporated. Kazi further discloses: *wherein said stub start, stub end, skeleton start, and skeleton end log data are used to determine a causality relationship data for a plurality of threads* (Page 3, first bullet).

As per claim 16, the above rejection of claim 9 is incorporated. Kazi further discloses: *wherein said stub start, stub end, skeleton start, and skeleton end log data are used to determine a causality relationship data for a plurality of threads spawned during invocation of said second software component* (Page 3 first bullet).

As per claim 17, the above rejection of claim 9 is incorporated. Kazi further discloses: *wherein said stub start, stub end, skeleton start, and skeleton end log data are used to determine a causality relationship data for a thread in which said first software component is invoked* (Page 3 first bullet).

As per claim 18, the above rejection of claim 9 is incorporated. All further limitations have been addressed in the above rejection of claim 5.

As per claim 19, the above rejection of claim 9 is incorporated. Kazi further discloses: *wherein the method includes a transportation of at least a portion of said stub start log data of said instrumented stub to said instrumented skeleton* (Bottom of page 8, e.g. "unique identifier").

As per claim 21, the above rejection of claim 9 is incorporated. Kazi further discloses: *wherein said instrumented skeleton stores at least a portion of said skeleton start log data to a thread-specific storage* (page 4 third paragraph: "Invocations of the same method executed under different threads are distinguished from one another by their **unique thread identifiers**").

As per claim 22, the above rejection of claim 21 is incorporated. Kazi further discloses: *wherein an event number included in said at least a portion of*

said skeleton start log data is updated before being copied into said thread-specific storage (Page 9 paragraph 3 “time stamp”).

As per claim 23, the above rejection of claim 9 is incorporated. Kazi further discloses: *retrieving a thread-transportable log data from a thread-specific storage of a parent thread; transporting said thread-transportable log data to a child thread; adding a thread information about a child thread to said thread-transportable log data to form a child thread data; and recording said child thread data to a thread table of said child thread (page 12 paragraph 2).*

As per claim 24, the above rejection of claim 23 is incorporated. Kazi further discloses: *wherein said thread-transportable log data comprises a self thread identifier and optionally a function container identifier (page 4 paragraph 3), <and> distinguishing user-application generated threads from threads generated by an underlying component-based system runtime infrastructure (page 8 paragraph 2 describes filtering calls from members of the Java or Sun packages which indicates that the caller is a Java library method and distinguishes it from the distributed monitoring system.).*

Kazi does not expressly disclose a self thread identifier that distinguishes user-application threads from component-based threads.

However, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use Kazi's method filter with the thread

identifier to produce a unique thread identifier that includes a user/component method designation. One of ordinary skill would have been motivated to produce a complete listing of method calls while enabling filtering to organize results.

As per claim 25, the above rejection of claim 9 is incorporated. Kazi discloses plans to implement memory management debugging (top of page 2), it does not expressly disclose logging heap memory usage data.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to implement memory management logging. One of ordinary skill would have been motivated to debug the garbage collection routines that slow down large applications.

As per claim 26, the above rejection of claim 9 is incorporated. Kazi further discloses: *wherein a particular log data is recorded in a per-process log table* (page 5 paragraph 3).

As per claim 27, the above rejection of claim 9 is incorporated. Kazi further discloses: *wherein a particular log data is recorded on a per-thread basis* (page 7 last paragraph).

As per claim 28, the above rejection of claim 9 is incorporated. Kazi further discloses: *wherein a particular log data is stored in a persistent storage*

(page 7 last paragraph).

As per claim 29, Kazi discloses: *processing an accumulated log data and calculating a system behavior characteristic for one or more software components executing within said component-based software system* See page 5 paragraph 2:

The tree generation step analyzes the merged trace files to create an output file containing the dynamic execution tree for a given client or server program. This output file is used by the visualizer to display the call graph.

All further limitations have been addressed in the above rejections of claims 1 and 9.

As per claims 30 and 31, the above rejection of claim 29 is incorporated. All further limitations have been addressed in the above rejection of claims 14, and 1, respectively.

As per claim 32, the above rejection of claim 29 is incorporated. Kazi further discloses: *wherein said system behavior characteristic comprises a shared resource usage data* (Table 1, "Machine Name").

As per claim 35, the above rejection of claim 29 is incorporated. Kazi further discloses: *wherein said system behavior characteristic comprises a timing latency data* (Table 1: "Total time").

As per claim 36, Kazi discloses a computer system (Figure 3). All further limitations have been addressed in the above rejection of claim 1.

As per claim 37, the above rejection of claim 36 is incorporated. Kazi further discloses: *a memory capable of storing said stub start and end log data* (page 8 paragraph 2).

As per claims 38 and 39, the above rejection of claim 36 is incorporated. All further limitations have been addressed in the above rejections of claims 9 and 1, respectively.

As per claim 40, the above rejection of claim 36 is incorporated. All further limitations have been addressed in the above rejections of claims 1 and 4.

As per claim 41, the above rejection of claim 36 is incorporated. Kazi further discloses: *wherein said memory further includes a thread table adapted to store thread log data* (page 7 last paragraph).

As per claim 42, the above rejection of claim 36 is incorporated. All further limitations have been addressed in the above rejection of claim 28.

B. Claims 8, 12, and 43 are rejected under 35 U.S.C. 103(a) as being unpatentable over the combination of Kazi, Blumson, Delucia, and Tucker as applied to claim 7, 9, and 36, respectively above, and further in view of prior art of record U.S. Patent 5,522,073 to Courant et al. (hereinafter referred to as "Courant"). *[See herein pages 17-18]*

As per claim 8, the above rejection of claim 7 is incorporated. Kazi does not expressly disclose regular expressions.

However, in an analogous environment, Courant teaches using a customizable regular expression to limit the information (column 8 lines 26-32).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Courant's regular expressions in Kazi's visualization system. One of ordinary skill would have been motivated to select only the messages that are of interest in order to clearly display results and maximize log utility.

As per claim 12, the above rejection of claim 9 is incorporated. Kazi does not expressly disclose enabling and disabling logging.

However, Courant teaches enabling a logging operation by sending messages (column 11 lines 23-31).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Courant's teaching of sending messages to enable logging in Kazi's distributed monitoring system. One of ordinary skill would have been motivated to control logging capabilities to minimize system perturbation when logging is not needed.

As per claim 43, the above rejection of claim 36 is incorporated. All further limitations have been addressed in the above rejections of claims 28, 29, and 12.

- C. Claim 20 rejected under 35 U.S.C. 103(a) as being unpatentable over the combination of Kazi, Blumson, Delucia, and Tucker as applied to claim 9 above, and further in view of prior art of record U.S. Patent 5,146,593 to Brandle et al. (hereinafter referred to as "Brandle"). *[See herein pages 18-19]*

As per claim 20, the above rejection of claim 9 is incorporated. Kazi does not expressly disclose the use of additional parameters in data transportation.

However, in an analogous environment, Brandle teaches the use of additional parameters to pass information to a function defined in an interface definition (column 4 line 67 – column 5 line 6).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Brandle's teaching of parameters to transport information in Kazi's distributed monitor. One of ordinary skill would have been motivated to provide caller information that would permit the callee to more quickly perform a task. By providing information as a parameter, the callee would not need to further analyze a function call since the information would be readily available.

- D. Claims 33 and 34 are rejected under 35 U.S.C. 103(a) as being unpatentable over the combination of Kazi, Blumson, Delucia, and Tucker as applied to claim 29 above, and further in view of prior art of record "Unix Power Tools" by Peek et al. (hereinafter referred to as "Peek").

As per claims 33 and 34, the above rejection of claim 29 is incorporated. Kazi does not expressly disclose CPU usage data.

However, in an analogous environment, Peek teaches the use of the Unix "ps" command which provides statistics regarding CPU and memory usage. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Peek's teaching of CPU usage in Kazi's distributed monitor. One of ordinary skill would have been motivated to analyze the relative use of system resources in an effort to locate performance bottlenecks.

(10) Response to Argument

- A. The rejection of claims 1-7, 9-11, 13-19, 21-32, and 35-42 under 35 U.S.C. 103(a) as allegedly being unpatentable over Kazi et al., in view of Blumson et al., in view of Delucia et al., in view Tucker et al. is improper. *[See Appeal Brief filed 12/26/2006, pages 8-12]*

On page 9 at the bottom, Appellants essentially argue that Kazi's .jta file is a data file and therefore cannot be the "instrumented stub" as claimed and so defined in at least paragraph [0027] of the present application. However, the Final Action, mailed 6/6/06, on page 7 paragraphs 3 and 4, show that Kazi was not relied upon to disclose the instrumented stub. Rather, Blumson was relied upon to teach the use of such an instrumented stub (page 6 section 6.1, e.g. "insert instrumentation"). Therefore, the argument is moot and not persuasive. Moreover, it should be noted that the plain language of the claim merely calls for "recording a stub start log data including a global causal identifier in an instrumented stub," and does not require any specific manner as defined (see paragraph [0027] as noted above) and argued for.

On page 10, paragraphs 1 and 2, Appellants appear to argue that Kazi's method identifier creation is at the second software component (Jvm 2), and cannot be a global causal identifier since it is not "CREATED by component 1, and then TRANSMITTED to component 2" (emphasis in original). However, it should be noted that the plain language of the claims merely

Art Unit: 2192

call for “transmitting the global causal identifier from the first software component to the second software component.”

In response to Appellants’ argument that the references fail to show certain features of applicant’s invention, it is noted that the features upon which applicant relies (i.e., “the global causal identifier is CREATED by component 1” – see page 10 paragraph 2, Brief filed 12/26/06) are not recited in the rejected claims. Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Moreover, Appellants misrepresent the art by contending that Kazi page 9 paragraph 3 describes a method identifier that is given by an RMI module at Jvm 2 (see Brief, page 10, top paragraph). To the contrary, Kazi page 9 paragraph 3 recites the following:

The trace entries in the client and server profiles are shown in Figure 4 for the RMI call Warehouse_Instance.lookup() **from the client on Jvm 1 to the server on Jvm 2** in the example in Figure 3. The client entry indicates that it is an RMI call to Object 25 for Method 5 on the server running on Machine cs1p62b.cs.umn.edu, which in this particular example is the Factory_Warehouse.lookup method on Jvm 2. [emphasis added]

Kazi clearly shows that an RMI call using the method identifier is created at Jvm 1 (e.g. “from the client of Jvm 1 to the server on Jvm 2”). Page 8 of the final Office action mailed 6/6/06 cites Kazi page 9 paragraph 3 to show that Kazi uses global method identifiers at both the server and the client. Even arguably, Kazi does not appear to expressly disclose where this global identifier is created. However, this argument is moot since the claims do not require such a creation location of any identifiers.

Appellants appear to contend that Kazi’s identifiers are not used globally to provide “end-to-end call tracking” (see Brief, page 10, middle paragraph). However, Kazi discloses recording identifiers. See page 9 paragraph 3:

Art Unit: 2192

The client entry indicates that it is an RMI call to **Object 25 for Method 5** ... The corresponding entry in the server profile indicates that it is an incoming RMI call from the client on Machine csln3.cs.umn.edu through Port 4667 for **Method 5 on Object 25**. Thus, the **server-side record links back to the entry in the client profile**" [emphasis added]

This passage indicates that the identifiers are used in a global fashion to link the client and server profiles. Kazi further discloses using such identifiers to provide a call graph, i.e. call tracking (see page 3, 2nd paragraph from the bottom, under the heading "The JaViz performance visualization tool set", e.g. "execution tree that shows the caller-callee relationship of the method calls in the program"; also see bottom of page 4 "Postprocessing"). As such Kazi captures such call tracking through the use of global identifiers. Therefore, Appellants' argument is not persuasive.

At the bottom of page 10, Appellants argue on the premise addressed above, that Kazi and Tucker cannot be combined. However, as pointed out above, the RMI call originates at Jvm 1, so the premise that it originates at Jvm 2 is flawed and the argument is moot and unpersuasive.

On page 11 at the top, Appellants essentially argues that Kazi and Tucker's identifiers are different from a global causal identifier and were not designed for call causality tracking. As pointed out above, Kazi was designed for profiling and uses call graphs which provide call causality tracking (see abstract and page 2, 2nd full paragraph , e.g. "caller-callee relationships" also Figure 1 "call graph"). Appellants have not provided any technical explanation how Kazi's disclosed call causality tracking occurs without identifiers. To this end, Appellants' arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims

Art Unit: 2192

define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

On page 11 at the bottom through the top of page 12, Appellants essentially argue that the citation to Kazi's Visualizer (as provided in 6/7/04 action, page 7) does not provide "log data contents is configured *during* operation" as recited in claim 7. However, while Kazi's Visualizer is a post processing tool, it uses log data that could only have been acquired *during* operation. The .jta file used by the Visualizer is created from .mrg files which are created from .prf, .svp, and .clp files, which are configured during operation. See Kazi Figure 1, also page 4 paragraph 3, and page 5 paragraphs 1 and 3 for further details. Therefore, Appellants' arguments are not persuasive.

B. The rejection of claims 8, 12, and 43 under 35 U.S.C. 103(a) as allegedly being unpatentable over Kazi et al., Blumson et al., Delucia et al., Tucker et al., and Courant et al. is improper. *[See Appeal Brief filed 12/26/2006, pages 12-13]*

On page 12 at the bottom through the top of page 13, Appellants argue that Courant teaches adjusting regular expressions *prior* to operation, not *during* as recited in claim 8. To further clarify the rejection, it should be noted that Kazi discloses configuring log data during operation (see the rejection of claim 7 addressed above). Also, Kazi discloses filtering trace data *during* operation (See page 4 paragraph 4), thus Courant is not relied upon to teach this feature.

Art Unit: 2192

But Kazi does not expressly disclose using regular expressions to provide the filtering. Courant teaches the use of regular expressions to filter data. See column 8 lines 26-32 supported by column 4 lines 57-61. Hence, Courant is not relied upon to teach “during said operation,” since Kazi discloses it by the use of filters as set forth in the rejection of claim 8. Accordingly, Appellants’ argument is moot and therefore not persuasive.

- C. The rejection of claim 20 under 35 U.S.C. 103(a) as allegedly being unpatentable over the Kazi et al., Blumson et al., Delucia et al., Tucker et al., and Brandle et al. is improper. See Appeal Brief filed 12/26/2006, pages 13-14.

All argument presented are based on prior arguments addressed above, and are not persuasive for the reasons set forth above.

- D. The rejection of claims 33 and 34 under 35 U.S.C. 103(a) as allegedly being unpatentable over the Kazi et al., Blumson et al., Delucia et al., Tucker et al., and Peek et al. is improper. See Appeal Brief filed 12/26/2006, page 14.

All argument presented are based on prior arguments addressed above, and are not persuasive for the reasons set forth above.

(11) Related Proceeding(s) Appendix

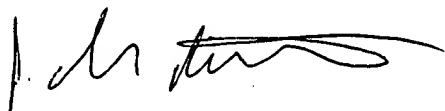
No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

Art Unit: 2192

For the above reasons, it is believed that the rejections should be sustained.

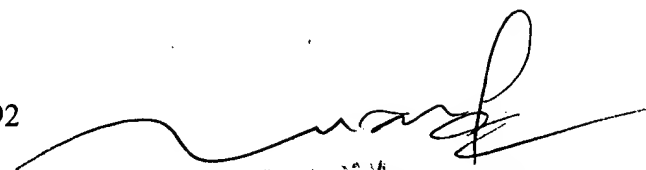
Respectfully submitted,

J. Derek Rutten



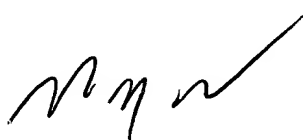
Conferees:

Tuan Dam, SPE AU 2192



SUPERVISORY PATENT EXAMINER

Wei Zhen, SPE AU 2191



WEI ZHEN
SUPERVISORY PATENT EXAMINER